Fake News Data Classification and Exploration

Finding text patterns in news articles for determining factual or fake information

Fahad Ibrahim

Student, George Mason University, f.ibrahim047@gmail.com

Saim Sandhu

Student, George Mason University, saimsdhu@gmail.com

Fake news has become a prevalent issue on the internet and various attempts have been made to detect it. Recent work has been done using Long short-term memory (LSTM) and GloVE embedding using a Kaggle dataset of labeled real and fake news articles. [4] This report revisits the dataset, comparing the performance of LSTM, Logistic Regression, and Decision Trees, and seeking to identify any biases. The results of Logistic Regression were favorable over the LSTM model, and both outperformed Decision Trees overall. However, a bias towards certain most frequent words in the text was found that make the dataset ill-suited for purposes extending beyond the scope of recent US presidential politics.

Additional Keywords and Phrases: LSTM, Logistic Regression, Decision Trees, EDA, bias, text classification, natural language processing

ACM Reference Format:
Saim Sandhu, Fahad Ibrahim. 2020. Fake News Data Classification and Exploration: Finding text patterns in news articles for determining factual or fake information.

1 Introduction:

Fake news is an increasingly noticeable issue in our growing online society and lives. The internet has allowed everyone in the world to learn, communicate, and express their ideas. However, the apparent drawback with this high accessibility is that there is no filter to differentiate factual and non-factual information. Any individual or group can publish fraudulent work without much repercussion. The obvious way to combat this is to be able to differentiate between fake and real information. Our goal in this report is to compare the performance of LSTM, Logistic Regression and Decision Tree Classifier in fake news detection and expand on the Exploratory Data Analysis (EDA) found in [3] by understanding other distinguishing characteristics of the texts and finding sources of bias in this dataset. We will first explain our model pipeline in three sections.

2 Model Pipeline

2.1 Pre-processing

The first task is to clean the dataset. The dataset was obtained from Kaggle and consists of two csv files, one that contains 23,502 fake news samples and another that contains 21,417 true news articles. Each text was appropriately preprocessed to extract useful information, using the NLTK and "re" python libraries. The titles and text were joined, and unnecessary columns, such as subject and date were dropped. HTML tags, emails, URLs, emotion symbols, and numbers were removed. Each review was then split into a list of its individual word

tokens. Stop words (insignificant words such as "I", "the") were removed. The words were lemmatized to their roots, allowing the feature count to decrease. The word tokens were finally combined into a single string.

2.2 Data Transformation

After cleaning the text and reducing it to its primary components, it is necessary to vectorize the text so it can be used by the model. Another issue is that not all the features in the text are relevant or useful, so keywords must be separated from irrelevant words. For this part we have chosen to use Keras Tokenizer and Word Embedding.

Keras Tokenizer first creates a word index, which is a dictionary of all the words with each word having a unique integer value based on frequency. This is used to convert the text into an integer sequence. Then padding is added to all texts to make them the same length for inputting into the LSTM model. [5] The parameters used: num_words = 1000 (Top 1000 words in from the text), max_len = 300 (max length of the text, basis for adding padding)

Word Embedding is a type of word representation where words with similar meaning have a similar representation. Each word is mapped to a real-valued vector, learned in a way that resembles a neural network. [1] We use the GloVE algorithm, as our online research shows that it is a superior method since it uses a co-occurrence matrix (words x context) that basically counts how frequently a word appears in a context. This allows it to predict the context of a word. In addition, we decided to use the twitter glove 100d pre-trained word embedding file. [4]

2.3 Model Execution

2.3.1 Long short-term memory (LSTM)

Long short-term memory (LSTM) is a type of artificial recurrent neural network architecture used in the field of deep learning. It can process single data points as well as sequences of data. We considered this model as we believed there existed semantic text sequences in fake and real news which could be detected with this model. The model structure includes an Embedding Layer, which includes the embedding matrix (twitter GloVE 100d), Dropout of 0.3 (dropping certain nodes to prevent overfitting), an LSTM layer (main layer for detecting data sequences), and a Dense layer (activation layer using sigmoid function). The results using different parameters are shown below.

Table 1: LSTM parameter testing

Model Iteration	1	2	3	4	5	6
training Size	1000	1000	1000	5000	1000	1000
num_words	1000	1000	1000	1000	5000	1000
max_len	300	300	300	300	300	500
Batch Size	64	256	64	64	64	64
Epoch Size	10	10	20	10	10	10
Testing Accuracy	0.84	0.83	0.86	**0.92**	0.89	0.86

2.3.2 Logistic Regression

Logistic regression is a supervised classification algorithm that predicts the probability of a target label. For our testing we are using a binary logistic model, for predicting the probabilities of the fake and real news. Our initial testing showed poor results. We then switched out the Keras tokenizer with a CountVectorizer in our data transformation segment and our results greatly improved. The results using different parameters are shown below.

Table 2: Logistic Regression parameter testing

Model Iteration	1	2	3	4	5	6
Sample Size	1000	1000	1000	5000	10000	20000
Max_features	100	500	1000	100	100	100
Testing Accuracy	0.976	0.976	0.964	0.987	0.990	**0.993**

2.3.3 Decision Trees

Decision Trees is another supervised learning algorithm that continuously splits the data based on certain parameters forming a tree like structure, where the leaves of the tree determine the classification of the data. Just like the Logistic Regression Model from before, we again faced issues with the Keras tokenizer and switched to using the CountVectorizer. The results using different parameters are shown below.

Table 3: *Decision Tree parameter testing*

Model Iteration	1	2	3	4	5	6
Sample Size	1000	1000	1000	5000	10000	20000
Max_features	100	500	1000	1000	1000	1000
Testing Accuracy	0.727	0.797	0.917	0.901	**0.921**	0.904

3 Exploratory Data Analysis (EDA)

In this section we will be carrying out an in-depth statistical analysis of the dataset to uncover characteristics or patterns in the text data that may be affecting the model performance in undesirable ways. Exploratory Data Analysis provides various techniques for analyzing a dataset. We look at the Class Balance and Word Frequency. [3]

2.1 Class Balance

This consists of comparing the distribution of the target labels of the data samples. This is important as an unbalanced dataset can give the illusion of high accuracy when the model is actually oversampling one target label and under sampling another. However, Figure 1 shows that our cleaned dataset (after preprocessing) is mostly evenly balanced between the fake and real news samples, so our model is mostly likely not suffering from oversampling or under sampling.

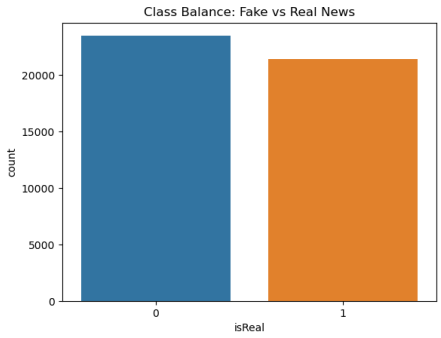

Figure 1: Fake vs Real Data Count

2.2 Word Frequency

In this part we will observe the most frequent words present in the fake and real news samples. This is important because most of the models we have used rely on processing single data points, except for the LSTM model which processes both fixed data points and sequences of data points. Due to this characteristic the models can be biased towards a few words, based solely on their excessive frequency. This tendency for a small set of words to cause bias was shown in [2]. Both Figures 2 and 3 show that there is an uneven distribution of words in the text with the top few words being significantly more frequent than all other words. This is problematic as the models could be relying on these few words for classifying the data. To test this suspension, we decided to run our models again except limiting our text data to the top 10 words and removing all others.

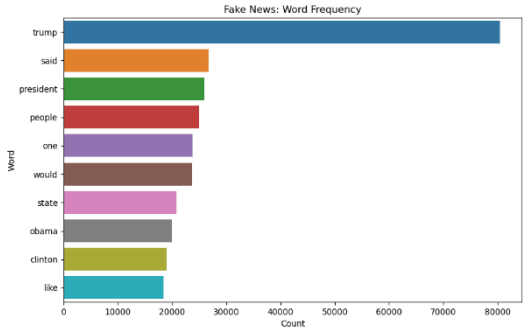

Figure 2: Fake News top 10 most frequent words

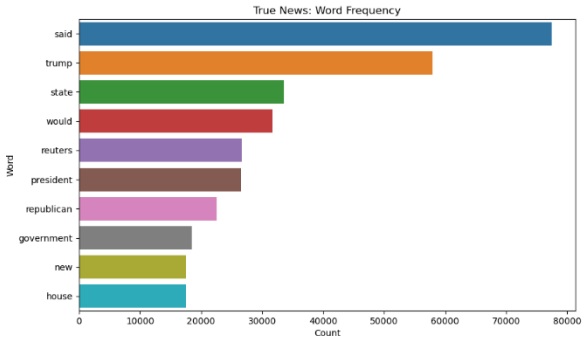

Figure 3: True News top 10 most frequent words

Table 4: *Testing accuracy with only top 10 words*

Model	LSTM	Logistic Regression	Decision Tree
Testing Accuracy with best Params	0.92	0.993	0.921
Testing Accuracy with top 10 words	0.80	0.78	0.72

The results shown by Table 4, show that even with only the top 10 words being kept in the text, the models are still performing well above 0.5. This means that the models are biased towards these words. This is problematic as these words are mainly representative of recent US politics, which is evident as the majority of the news articles have been taken from recent years of US politics. Therefore, the model may not do well with other news types.

Conclusion:

Overall, we were able to reach our goals of comparing the performance of different models on fake news detection, understand distinguishing characteristics of fake news, and finding sources of bias. In comparing the models, we were initially surprised that the Logistic Regression and Decision Tree models outperformed the LSTM model. Later in the EDA part, we discovered this was because there was an uneven distribution of words in the dataset, and the models were being biased towards the top frequent words. The solution to this bias is to produce a more balanced dataset not just in target labels, but also in themes, topics, regions, and historical events. Until this is done the current accuracy of the models will not matter much, as they will perform poorly on other news datasets.

REFERENCES

[1] Jason Brownlee. 2019. What Are Word Embeddings for Text? (August 2019). Retrieved December 13, 2020 from https://machinelearningmastery.com/what-are-word-embeddings/

[2] Josutk. 2020. Only one word 99.2%. (August 2020). Retrieved December 13, 2020 from https://www.kaggle.com/josutk/only-one-word-99-2

[3] Andrii Shchur. 2020. Fake news detector with deep learning approach (Part-I) EDA. (July 2020). Retrieved December 13, 2020 from https://towardsdatascience.com/fake-news-detector-with-deep-learning-approach-part-i-eda-757f5c052

[4] madz2000. 2020. NLP using GloVe Embeddings(99.87% Accuracy). (July 2020). Retrieved December 13, 2020 from https://www.kaggle.com/madz2000/nlp-using-glove-embeddings-99-87-accuracy/data

[5] Real Python. 2020. Practical Text Classification With Python and Keras. (November 2020). Retrieved December 13, 2020 from https://realpython.com/python-keras-text-classification/

Presentation Video Link: https://youtu.be/t_stySbXFPM